# A DISTANCE LEARNING ENVIRONMENT ARCHITECTURE

Lorenzo SOMMARUGA, Nadia CATENAZZI
MEDIATECH S.r.l.
C.P. 100, 09018 Sarroch (CA), Sardinia (Italy)
l.sommaruga@fantastic.com, nadia.catenazzi@bigfoot.com

Sylvain GIROUX, Claude MOULIN
CRS4
Loc. Macchiareddu, Uta (CA), Sardinia (Italy)
sgiroux@crs4.it
claumou@ciaoweb.it

**Abstract**

This paper presents the core architecture of NURAXI, a multimedia research platform aimed at the design, generation, deployment, management and use of intelligent distance learning environments. It describes the actors and services that are involved in the production and use of intelligent distance learning environments, the structures that are at the heart of the environment, and the processes that use these structures. The competence notion is the central component around which both author's and student's interactions gravitate. An author generates the training material on the basis of competencies to be acquired by a student, and the means to get them, i.e. the related activities and contents. A student accesses the pedagogical material by first selecting his/her learning objectives. The training path is made of dynamically created pedagogical activities. Examples of implemented activities are also shown in the paper. The platform presents a number of interesting features including re-usability of didactic components; adaptability of the training material to the student model; dynamic definition of the training path; modularity, and interoperability thanks to the adoption of standard and open solutions in terms of document structures, ontologies, design and implementation techniques. All these advantages derive from the integration of technologies such as the XML paradigm, Servlets, Software Agents, and Distributed Databases.

**Keywords**: Distance learning architecture, pedagogical activities, document typology, XML.

## Introduction

Internet, the web and artificial intelligence disclosed new opportunities and new ways to train people at distance. In addition, the demand is there for such services. The fast evolution of today world impels corporations to provide their employees just-in-time training, adapted training and more generally continuous education. On the other hand, the offer (authoring) has to cope with fast evolving knowledge. Thus to be able to satisfy the demand, authors need means to specify once didactic material and then to reuse, select, adapt and distribute this material to different users in different contexts. In some sense, they need intelligent authoring tools to achieve a sort of just-in-time opportunistic authoring, i.e. providing the didactic material at the very end of the process when the learner needs it, just like in a true dialogue.

Since September 1998, Mediatech is developing NURAXI with this goal in mind. NURAXI is a multimedia research platform aimed at the design, generation, deployment, management and use of intelligent distance learning environments. The educational model underlying NURAXI relies upon the notion of dialogue. Like in a real dialogue, NURAXI tries to select, generate and adapt the information to be delivered to the current understanding, the learning style and the objectives of the learner. To achieve this, NURAXI interacts with the user on the basis of the competency assessment (initial, on-going progress and final), the individual learning style and collaborative learning. The reader used to intelligent tutoring system may think that there is indeed nothing new under the sun. However, our solution is departing from a document type based organization of courses and training material towards a functionality and competency based model. This means breaking with the traditional view about the structure of the teaching material. We consider models and structure for information, knowledge and competencies more appropriate to the new on-line delivery environment than the document-based old one (html-based or not). From a technological point of view, this was made possible by the recent arrival of XML technology and its coupling with Java.

This paper presents the core architecture of NURAXI. More precisely, it describes the actors and services that are involved in the production and the use of intelligent distance learning environments, the structures that are at the heart of NURAXI, and the processes that use these structures. In addition, it provides some examples of implemented pedagogical activities to show how reusability and adaptability could be practically achieved.

## Services and Actors in NURAXI

### Services

NURAXI is a very flexible platform. It is built in such a way that the borderline, which separates tasks done by humans from tasks done by machines, is a moving one depending on the situation and on the people needs. We use the notion of service to model this moving frontier. You can see services in NURAXI as points where humans can take the control in order to inject more sophisticated information in the system. At the same time, these services release authors and learners from complex tasks. For instance, an author can decide to select solely the competencies he/she wants to teach, letting NURAXI decide the pedagogical activities. Alternatively, the author can decide the competencies and precisely specify a path through pedagogical activities.

NURAXI supplies many services including:

- effective individualized courses based on competency models, ontologies for documents, learning styles, teaching strategies, adaptive interfaces, and dynamic computation according to previous actions
- dialogue and communication support in groups by shaping the high-level collaborative learning environments according to the collaboration and conversation models
- didactic material creation and adoption using DTD as templates coupled with XML and XSL
- assemble and reuse of course material using JavaBeans, XML (meta data), and XSL
- updating of course material using the Web and on the fly computations
- management of huge amount of documents using XML and distributed databases
- the management of competencies, contents, activities and knowledge.

### Actors

The platform is intended to be used by different actors:

- the author, who is responsible for creating a course, pedagogical activities and didactic contents;
- the student, who aims to acquire new knowledge and competencies;
- the guest, someone interested in getting general information about courses, methodology of teaching, course curricula;
- the administrator (including the backoffice and system administrator), responsible for the administration tasks and for the system management;
- the librarian, in charge of organising and maintaining the library;
- the tutor, who plays the role of instructor, advisor, or facilitator in the virtual classroom.

## Core Structures

The NURAXI platform architecture is composed of various modules corresponding to actors and functionalities involved in the learning process. The two main actors of this process are the author and the student of a course. Every action and interaction in this process occurs around the same core elements that form both the basic structures of the teaching material and the basic infrastructures for the learning process. The author and the student, as described in the Processes section use these common structures in a different way.

The competence notion is the central component around which both author's and student's interactions gravitate. In fact, an author generates the training material on the basis of learning objectives, i.e. competencies to be acquired by a student through a training process, and the means to get them, i.e. the related activities and contents. A student accesses the pedagogical material by first selecting his/her learning objectives.

The main structures introduced in NURAXI include: competence, knowledge, course, pedagogical activities, contents and student model.

### Competence and knowledge

A competence is the ability to do something well or effectively. It can generally be considered as a set of "savoirs, savoir-faire, savoir-être" (knowledge, know-how, and attitudes) which are activated at the accomplishment of a given task [Cognitivo2 1999]. In particular, in our pedagogical context, competence is an abstract concept which can be reified through attributes or properties that qualify and quantify the concerned ability.
A competence is a potential act associated with the performance of an action or task.

Within NURAXI competencies are classified according to the domain area (e.g. Computer Science, Languages, Mathematics) and to Bloom's learning outcomes (Knowledge, Comprehension, Application, Analysis, Synthesis and Evaluation) [WestEd 1999, CTE 1999].

A competence is represented by means of an identifier and described, in short, by the *what*, *how*, *how much*, *when* and *why* interrogatives.

- The *what* question is the core of the competence definition. It defines the subject of the act in the form of a *verb* and an *object*. The *verb* describes the action of the competence. The *object* specifies

the context of execution of the verb. Both *verbs* and *objects* are grouped into dictionaries and classified according to the Bloom's taxonomy and the domain area.

- The *how* question concerns the means and/or procedures to be used to perform the act. It may be a description of the procedure to follow in order to accomplish the act and to perform well, effectively or efficiently.
- The *how-much* question relates to the qualitative and/or quantitative degree of an act. It can be expressed by means of adverbs or adjectives that better qualify and/or quantify the expected result of the act. It can be used to give an expected and/or performed measure of the act performance.
- The *when* relation may be useful to state some temporal requirements or preconditions that are necessary in order to correctly accomplish the act.
- The *why* relation (causal) provides information about the reasons that may imply the need for the act accomplishment. It can be useful to search for necessary competencies given a particular problem or need.

An example that shows the use of these interrogatives is the definition of the following competence:

| | |
|---|---|
| Verb: | Creating |
| Object: | HTML pages |
| How: | Using an HTML editor |
| How much: | graphical and animated |
| When: | Being able to use an editor |
| Why: | To develop a web site |

Competencies can be simple or composed. The composition of competencies is expressed through the *part-of* relation, which allows a competence to be split into simpler sub competencies.
All the defined competencies are collected in a *competence dictionary*. A graphical representation of the dictionary may be drawn in the form of a graph, the *competence graph*, where nodes are competencies and edges are the *part-of* and *when* relations.
Other important structures complete the competence description, including the related knowledge, associated activities, and the *adaptive standard level* of competence.
When actuating a competence to perform a task an individual makes consciously or unconsciously use of a certain amount of information and knowledge. In our system this knowledge is explicitly related to the competence and stored in the knowledge level in the form of a graph.
A link to the activities that are useful to acquire a competence is maintained. The related activities are detailed and stored at the activity level.
The standard level of competence allows a competence to be adaptively assessed. It is composed of a standard level distribution and a minimal level of acquisition. The standard level distribution may be represented as a distribution function of the learner performance levels. This is computed and updated on the basis of the assessed performances of all the learners (see competence level in the student model). The minimal level of acquisition represents the threshold of acceptance for the competence performance. It could be a single value within the distribution or a more complex description such as a weighted measure of activity performances.
The dynamic adaptation of the standard level allows for a more relative measure of the learners' competence according to their performance. This is opposed to the traditional view of an absolute comparative measure of the competence level, according to a tutor.

## Course

A course is defined by an author as a list of learning objectives which aim to a particular pedagogical goal. The learning objectives are the same for all the students attending the same course; therefore they may represent a predefined shareable set of learning objectives. Thanks to this nature, a course can be certified, assuring in this way the individual ability in the specific field. A course is an induced subgraph[1] of the *competence graph*.
The course structure also contains information such as the list of competencies that must be the training target for the student; the audience to whom the course is addressed; a description of the certification that is obtained; the authors of the material, and the creation and last updating dates.

## Pedagogical Activities

Pedagogical activities are associated with each learning objective that they aim to reach.
At the highest level, activities have been classified into two broad categories: individual and collaborative activities. Individual activities have been also divided in various groups including:

- answering (a question)
- solving (a problem),
- searching (for information),
- writing (a report, a text, an essay, a message),
- using (a simulator)
- exploring (a world),
- attending (an example, a counter-example, a theory item, a story).

Examples of collaborative activities include:

- Jigsaw,
- brainstorming,
- debates,
- consensus building.

[1] An *induced* (generated) *subgraph* is a subset of the vertices of the graph together with all the edges of the graph between the vertices of this subset.

This classification is not intended to be exhaustive, but it represents a possible starting point. It is based on other similar works [McConnell 1999, Dalgarno 1999]. For instance, Dalgarno proposes learner activities likely to facilitate the achievement of specific learning outcomes. Our purpose is to start from this limited set to prove the soundness and effectiveness of the layered model.

A number of properties are associated with each activity in order to specify:

- Learning outcomes: they are classified, for instance, according to the Bloom's Taxonomy;
- Learning style: it indicates the most suitable learning style. It does not prevent the same activity from being used for other learning styles;
- Required tools: it refers to the tools which are necessary to accomplish or assess that activity;
- Competence objectives: it indicates the list of the competencies the activity contributes to acquire;
- Complexity level: defined by the author, it is a measure of the difficulty level of the activity;
- Contents: it refers to the contents used to create the activity (see next section).

### Contents

The contents represent the didactic material and constitute the basic bricks of the NURAXI platform. The contents are used to generate the activities presented to the student, and may be used in different ways in several activities. Contents can be created by authors or imported from external sources and integrated into the system by means of meta data.

All the didactic material is represented in the form of *Compacted content.*
This element mainly contains information such as the list of activities where the content is used, links to reference material used to support the argument, *content* or *integrated content*, and a meta data description of the material.

The *content* represents the underlying structure of the didactic material and it allows the same information to be shown differently according to the activity where it is used. For instance a common content can be used both as an exercise and as an example. We assume it is formalised in XML.
The *integrated content* consists of already existing material, which can be available in any format, created outside the platform.
The meta data description creates a meta level similar to the IMS meta data specification [IMS 1999] and based on RDF [W3C 1998] that is used mainly to integrate existing material but also for XML documents.
Additional information about the date and authors of the contents is also provided.

### Student Model

The student model is made of a static and a dynamic part. The static part contains information such as personal data. The dynamic part contains information about the initial learning objectives, the learning style, the learning status of the student, and the competence level.

- Personal data include information about the student such as name, surname, address, age, etc.
- The initial learning objectives indicate the competencies the student intends or has to acquire. If the student has selected an entire course, the associated set of learning objectives could be shared among different students. In this case this set could be stored once and re-use many times.
- The learning style of the student indicates how the student naturally learns. It is not static, but could be modified during the training process. The selected learning style model is inspired by the Meyer-Briggs personality types (MBTI, 1999).
- The learning status contains information about the path followed by the student in terms of accomplished activities, result of the activity evaluation, and *competence level* for each considered competence.
- The competence level is computed on the basis of the activity evaluation, and it is compared to the minimum level of acquisition to assess whether the competence has been satisfactorily acquired. The competence level reached by the student updates the standard level distribution of that competence.

## Processes

### The authoring process

The author is responsible for creating a course. The authoring process is supported by services that the NURAXI platform provides in the form of visual tools, such as competence, activity, and content editors.

As already mentioned a course is defined in terms of learning objectives, i.e. the competencies to be acquired. In order to guarantee re-usability, a central competence dictionary is maintained. The author will select a list of competencies for a course from the competence graph. If the competence the author wishes to select is not available in the graph, he/she may add it, extending the graph.

Once the learning objectives are defined, the author will select a number of pedagogical activities that the student may undertake to get that competence and related knowledge. If the selected competence has already some associated activities, the author may decide to use them for the course or to add new ones if

he/she considers that the existing one are not appropriate. The task of creating new activities involves the selection of an activity type (for instance answering a question, solving a problem) and the creation or integration of didactic contents. The author may define simple activities or composed activities. Composed activities are collections of activities to be shown in sequence or at the same time, according to predefined models of composed activities.

If the content is created from scratch, a possible procedure consists of filling first the content related to the selected activity. We assume that the same contents, appropriately filtered, can be used in many activities. Therefore the system prompts the author for other information that can complete that content allowing other activities to be semi-automatically defined. This could be easily achievable if the content is formalised in XML. If the author intends to integrate existing content into an activity, he/she will provide meta-information and a reference to this content. The choice to include didactic material and integrate it into the platform through meta-data allows existing material, widely available in Web, to be re-used in its original form without converting it to a predefined format.

## The learning process

The student is an actor who uses the platform with the purpose to reach some learning objectives, i.e. to acquire new competencies. These learning objectives can be associated to a course (in this case a certification can be obtained) or be selected independently from a course by the student.

The most interesting aspect of the environment is that the training path can be dynamically created. This means that there may be no predefined paths the student has to follow, and no static activities. The training path is a trail through pedagogical activities that are created on the fly by combining various contents. The path is determined on the basis of the student learning objectives, competence level, and learning style.

The selection of the next activity may be accomplished in different ways: it may be decided by the system, by the student guided by the system, or on his/her own.

If the selection of the next activity is done by the system, it is a function of the student learning objectives, learning style, competence level, and history of the interaction with the system. All this information is maintained in the student model. Every activity accomplished by the student is evaluated; by combining the result of evaluating the activities associated to a learning objective, it is possible to compute the competence level reached by the student.

## Some examples of implemented activities

In the context of the architecture presented above we mainly concentrate on activities and content formalization. In particular three activity types have been implemented: *answering*, *solving*, and *attending*. For each of them a model has been defined, and formalised in XML by means of a document type definition (DTD).

For *answering (a question)* activity type the *test* DTD has been defined (see figure 1). Different types of questions are foreseen including: multiple choice, true or false, short answer, matching, fill-in the blank, etc. A multiple choice consists of a question and a number of answers. One or more hints may be associated to the question or the answers. Various feedback may be associated to the question or the answers. Hints and feedback are optional elements.

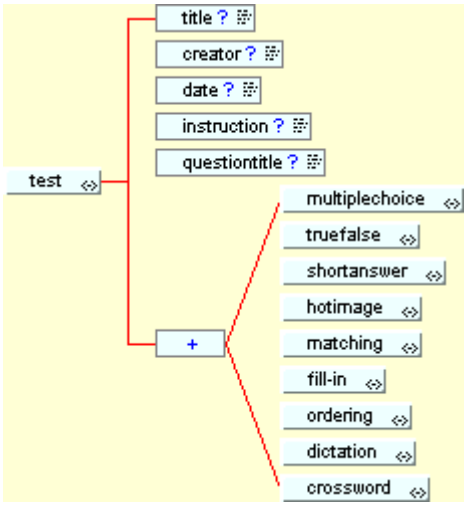


Figure 1: Test.dtd

Figure 2 and 3 show some examples of how the test activity could be presented to the student, focusing on the multiple choice question type.


Figure 2: an example of test activity

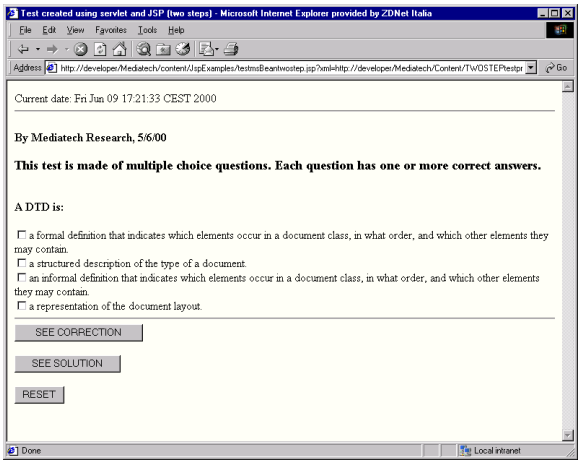


Figure 3: another example of test activity

The *solving (a problem)* activity has been formalised with the *problem* DTD. It mainly consists of parts containing a question and one or more solutions to this question, as illustrated in figure 4.

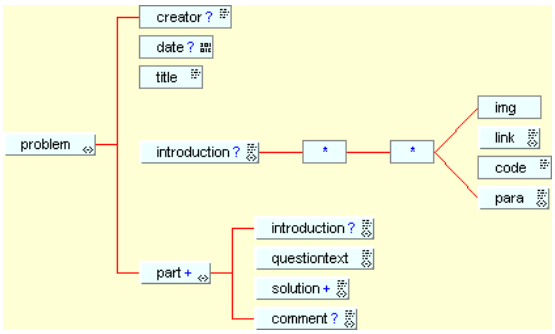


Figure 4: the problem DTD

Figure 5 shows an example of how a solving activity could be interactively presented to the student.

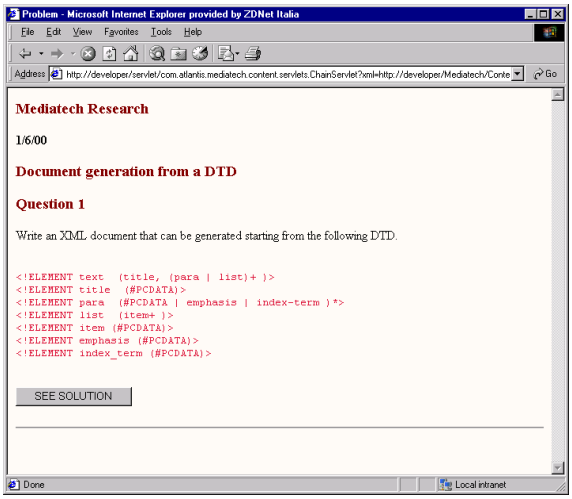


Figure 5: problem solving activity

The third type of activity that has been formalised is the *attending* type. Figure 6 shows the corresponding DTD. With respect to the previous examples, this activity is less interactive. Its purpose is to present a concept by means of definitions, examples, counter-examples, procedure, etc.

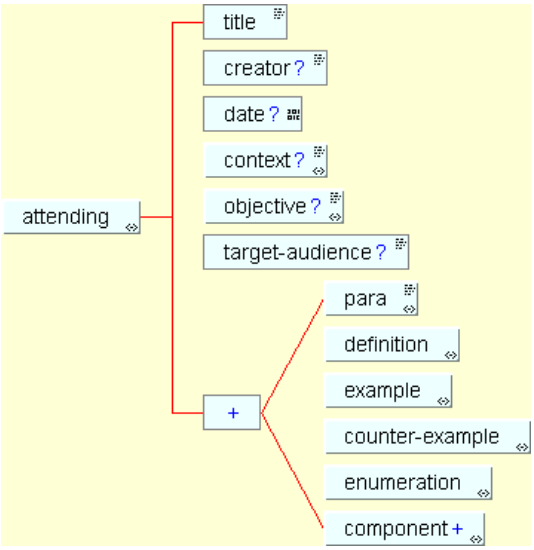


Figure 6:the attending DTD

Figure 7 shows how the attending activity could be presented to the student.

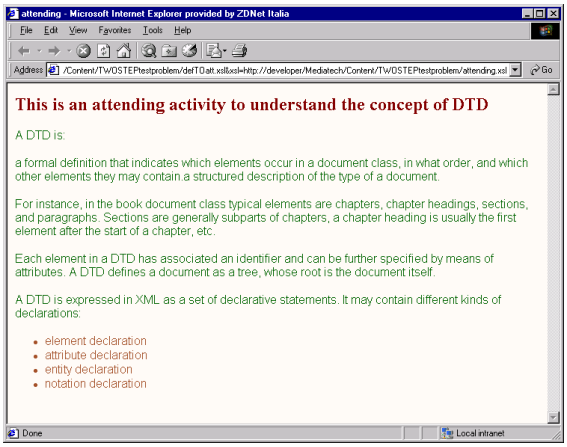


Figure 7: an example of an attending activity

As mentioned in the previous sections, contents are used to generate activities. The key idea is that activities are not based on static predefined contents created according to the models (DTDs) previously shown, but that the activity contents are dynamically created starting from other basic reusable contents.
Hereafter we will show some examples that will clarify this concept. Considering the similarities between a solving activity and an answering, just one document could be written that could be used to generate both the solving and answering activities, by appropriately filtering its contents.
An example of reusable content is shown in figure 8. This document has been used to produce both the problem shown in figure 5 and the multiple choice test shown in figure 2.

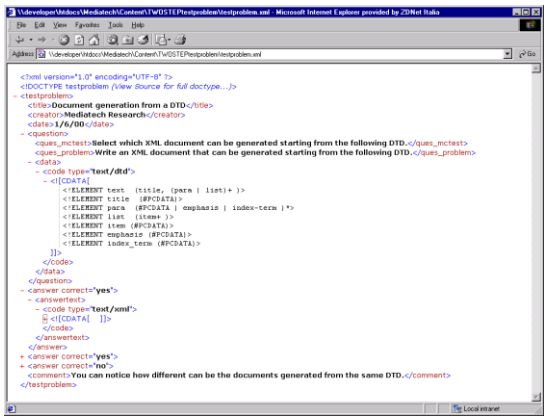

Figure 8: testproblem.xml

Another example is shown in figure 9. The def-DTD.xml document is a basic content that has been used to generate the test shown in figure 3 and the attending activity shown in figure 7.

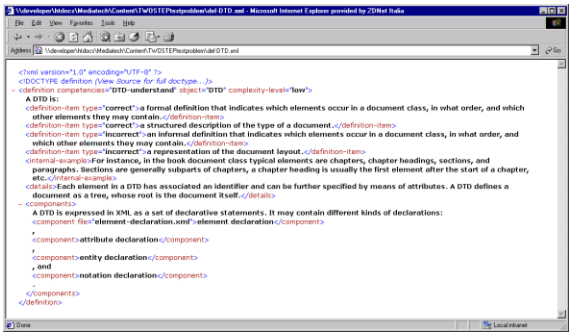

Figure 9: def-DTD.xml

The diagram shown in figure 10 summarised the process of dynamic generation of activities. The process consists of two steps:

- the first one consists of a *transformation* of some basic contents into the activities contents, i.e. XML contents based on the activity DTDs;
- the second step consists of *presenting* the activity to the student. It is important that the student is able to interact with the activity; therefore an activity can take different states corresponding to the different steps of the user interaction. In addition, the same activity could be presented to the student in a different way according to his/her model (adaptability).

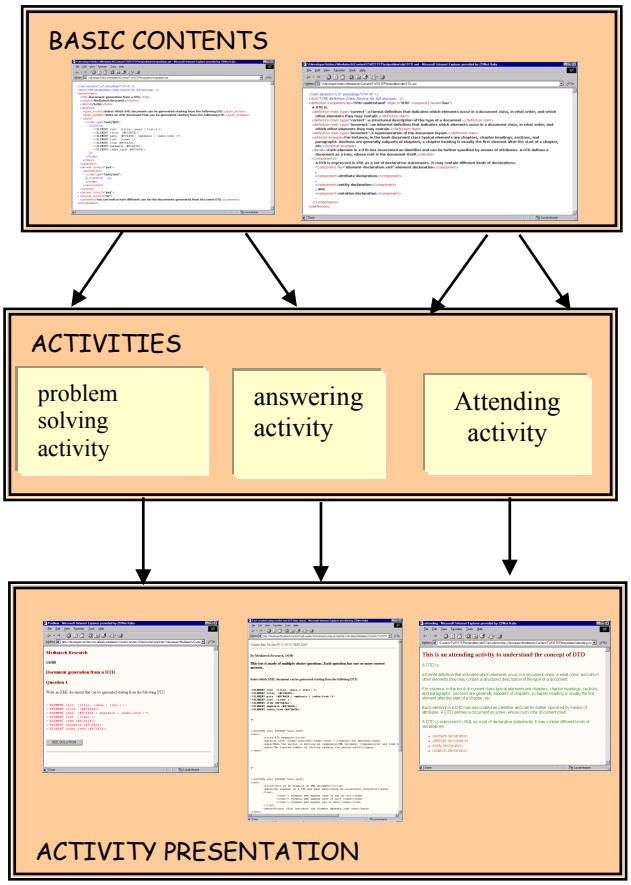


Figure 10: activity generation

From a practical point of view the transformation step has been accomplished by using the XSLT transformation language that converts the basic contents formalised in XML into the activity contents, also expressed in XML. The second step (presentation) has been implementing in different ways depending on the activity type. For instance for answering activity a *jsp* program has been written, that uses DOM to access the XML document structure and presents the activity as HTML pages. For the solving activity an XSL stylesheet is used. The composition of these two steps has been implemented using a servlet or a *jsp* program.

## Conclusions and developments

The learning environment described in this paper is designed to provide a number of features:

- Re-usability: a library of reusable didactic components (contents, activities, competence dictionaries) is created;
- Adaptability: the same material is presented in different ways according to the student learning objectives, learning style, competence level, and the history of interaction;
- Modularity: the architecture is designed in such a way that new modules could be easily integrated;
- Interoperability: we are going towards standard and open solutions in terms of document structures, ontologies, design and implementation techniques.

All these advantages are made possible thanks to the integration of technologies such as the XML paradigm, including DOM and XSL stylesheets, Servlets, Software Agents, and Distributed Databases. The NURAXI design is based on the UML modeling technique, and the implementation exploits JSP and Java Programming.

Some of these technologies have been already tested in various demonstrators developed to show particular functionalities or potential features of the NURAXI platform.

For instance two demonstrators have been implemented in order to show possible approaches to produce personalised hypermedia courses. The first one demonstrates how content, competency and student models can be effectively integrated using XML in order to produce personalised adaptive learning material [Catenazzi and Sommaruga 1999]. The other demonstrator indicates how to support multiple learning and teaching styles in order to provide dynamic adaptation and presentation of individualized learning material [Moulin 1999].

Another demonstrator [Todorova 1999], integrating database technology and portable Java code in a web-based distributed environment, aims to develop the basic infrastructure required to build and access on-line courses, and face administration issues such as student registration and course management.
Collaborative tools within a learning environment have been investigated in another demonstrator that shows a sample of a collaborative learning environment supporting various collaborative learning techniques such as Jigsaw, role playing, reciprocal teaching, etc. [Cenati and Sommaruga 1999].

On the basis of the previous considerations, it is possible to conclude that, according to Brusilovsky's analysis framework for adaptive technologies [Brusilovsky's 1998], the current state of NURAXI applies adaptive techniques in the following areas: curriculum sequencing (both knowledge and task sequencing) and adaptive presentation technology. In the near future, NURAXI will tackle adaptive collaboration support, intelligent analysis of student solutions and interactive problem solving support.

## References


Brusilovsky, P. 1998. Adaptive Educational Systems on the World-Wide-Web: A Review of Available Technologies. Workshop-Intelligent Tutoring Systems on the Web, ITS 98. http://www-aml.cs.umass.edu/~stern/webits/itsworkshop/brusilovsky.html

Catenazzi, N. and Sommaruga, L. 1999. "XML e l'apprendimento a distanza. XML Italia, I giornata di studio XML Italia, Bologna, june [http://www.xml.it/SlideBologna/slides.html].

Cenati, M., and Sommaruga, L. 1999. Interaction and Dialogue in the NURAXI On-line Education Environment: an Example of Collaborative Learning with Jigsaw, *Roles of Communicative Interaction in Learning to Model in Mathematics and Science, C-LEMMAS 1999, Ajaccio, Corsica, 15-18 April*

Cognitivo2 1999. http://www.fse.ulaval.ca/fac/dpt/cours/ten62630/Cognitivo2/pagemenuprinc.htm

CTE - Center for Teaching Excellence 1999. Bloom's Taxonomy
http://www.stedwards.edu/cte/bloomtax.htm

Dalgarno, B. (1998) Choosing learner activities for specific learning outcomes: A tool for constructivist computer computer assisted learning design. Edtech'98 Proceedings
http://wwwasu.murdoch.edu.au/aset/confs/edtech98/pubs/articles/abcd/dalgarno.html

IMS Meta-data Specification (1999) http://www.imsproject.org/metadata/index.html, August 1999

MBTI (1999) *MBTI® and Psychological Type* http://www.itd.net.au/aboutmbti.htm

McConnell, J.J. (1999) Activity List http://www-cs.canisius.edu/~mcconnel/activity_list.html

Moulin, C. (1999) Typology of shared documents in a Web-based learning environment, *Proceedings of the AIED'99 workshop on "Ontolgies for Intelligent Educational Systems*", pp. 58-65

Todorova, D. and Maraschi, D. (1999) Demonstrator 1: Infrastructure, Registration and DB2 Management, Mediatech Technical Report n° 73
W3C (1998). Resource Description Framework (RDF) Model and Syntax Specification - W3C Working Draft, October http://www.w3.org/TR/WD-rdf-syntax/

WestEd (1999) Bloom's Taxonomy http://www.wested.org/tie/dlrn/blooms.html